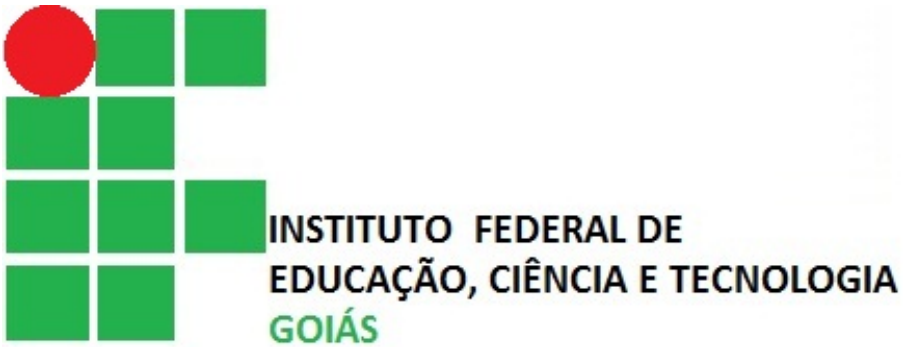
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS (IFG)

CAMPUS JATAÍ



CONTROLADORES LÓGICOS PROGRAMÁVEIS (CLP´s)

Linguagem de Lista de Instruções

Prof. Dr. André Luiz

1 – Introdução

A linguagem de Lista de Instruções (LI), também comumente referenciada pelo nome original da língua inglesa, Instruction List (IL), define mnemônicos (auxiliar de memória) como é feito na linguagem assembly utilizada nos microprocessadores e microcontroladores. Os mnemônicos representam operações lógicas booleanas e comandos de transferência de dados.

Em relação as demais linguagens, apresenta as seguintes características:

Vantagens

- Correspondência entre comandos da linguagem e as instruções assembly do CLP, facilitando a estimativa do tempo de execução do programa.
- Documentação mais compacta do que a equivalente com relés.

Desvantagens

- Necessidade de familiarização do operador com álgebra booleana.
- Necessidade de uma certa noção de programação em assembly.
- É normalmente difícil e trabalhoso realizar eventuais alterações no código já implementado.

A LI é a linguagem ideal para resolver problemas simples e pequenos em que existem poucas quebras no fluxo de execução do programa. É, portanto, particularmente adequada para CLPs de pequeno porte.

Essa linguagem pode ser usada para descrever o comportamento de:

- Funções;
- Blocos de funções;
- Programas

2 - Principos básicos

A linguagem de Lista de Instruções é semelhante ao código assembly com comandos load e store. Ela usa o conceito de acumulador para armazenar os resultados intermediários.

- Cada instrução utiliza ou modifica o valor de um único registrador denominado registro de resultado ou acumulador.
- As instruções são executadas no conteúdo do acumulador.

- O operador indica o tipo de operação a ser feito entre o resultado atual contido no acumulador e o operando.
- O resultado da operação é armazenado no próprio acumulador.

3 - Sintaxe

As regras principais de formação de um programa em linguagem de Lista de Instruções são:

- Cada instrução deve começar em uma nova linha
- Cada instrução pode ser precedida por um rótulo (elemento opcional) que é indicado com um nome seguido de dois pontos ":".
- Uma instrução é composta de operador e operandos (instrução = operador + operandos).
- O operador pode ou não incluir um modificador.
- Caso seja necessária a inclusão de mais de um operando, estes devem ser separados por vírgulas.
- Se for desejada a inclusão de comentário, ele deve ser o último elemento da linha.
- Um comentário é iniciado pela seqüência de caracteres (* e terminado pela seqüência*).
- Linhas em branco podem ser inseridas entre instruções.
- Um comentário pode ser colocado em linha sem instruções.

Esta estrutura pode ser verificada na Figura 1e no exemplo 1.

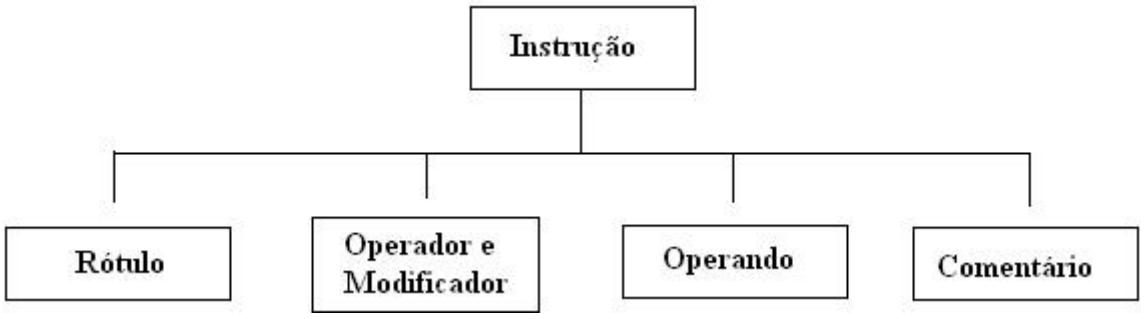


Figura 1 – Estrutura de uma linha de instrução da linguagem LI.

Exemplo 1:

Rótulo	Operador	Operando	Comentário
Início	**LD**	**% IX1**	**(* botão pressionado?*)**
	AND	**% MX3**	**(* comando válido*)**
	ST	**% QX1**	**(* liga o motor)**

O valor da entrada % IX1 é carregado para o acumulador, em seguida é feita um operação lógica AND entre o conteúdo do acumulador e da memória % MX3. O resultado é transferido para a saída %QX1.

4. Rótulo (etiqueta)

Cada instrução pode ser precedida por um rótulo, que é um nome seguido do caractere ":". Ele também pode ser colocado em uma linha que não contenha nenhuma instrução. Os rótulos são utilizados como operandos PR certas instruções tais como saltos. A sua nomenclatura deve obedecer a seguintes regras:

- O comprimento não deve exceder 16 caracteres.
- O primeiro caractere deve ser uma letra.
- Os caracteres restantes podem ser letras, números ou símbolo "_" (sublinhado).
- Não pode haver no mesmo programa dois rótulos iguais.

5 - Modificadores de instruções

A lista a seguir representa os modificadores permitidos para as instruções da linguagem. Devem ser anexados após o nome da instrução, sem caractere separador.

- N = inversão lógica do operando;
- (= operação adiada;
- C = operação condicional.

O Modificador "N" indica que o operando deve ser invertido antes de ser utilizado pela instrução. Por exemplo, a instrução ANDN % IX1 é interpretada como "o conteúdo de %IX1 é invertido e com o valor de resultado é feita ma operação lógica AND com o acumulador".

O Modificador abrir parênteses "**(**" indica que a avaliação da instrução deve ser adiada até que seja encontrado o próximo fechar parênteses "**)**".

O modificador "C" indica que a instrução deve ser executada somente se o conteúdo atual do acumulador tiver lógico verdadeiro (ou diferente de zero para tipos não booleanos). O modificador

"**C**" pode ser combinado com o modificador "N" para indicar que a instrução não deve ser executada, a menos que o resultado seja falso (ou 0 para tipos não booleanos).

A tabela 1.1 apresenta os principais comandos da linguagem de Lista de Instruções.

Operador	**Modificador**	**Operando**	**Descrição/ significado**
LD	N		Carrega o operando para o acumulador
ST	N		Armazena o conteúdo do acumulador no local especificado pelo operando
S		BOOL	Faz com que o valor do operando seja 1
R		BOOL	Faz com que o valor do operando seja 0
AND	N, (		Função booleana AND
&	N, (		Função booleana AND
OR	N, (		Função booleana OR
XOR	N, (		Função booleana OU - Exclusivo
ADD			Soma
SUB			Subtração
MUL			Multiplicação
DIV			Divisão
GT	(		Comparação (Greater Than) maior que (>)
GE	(		Comparação (Greater or Equal) maior ou igual que (>=)
EQ	(		Comparação (Equl to) igual a (=)
NE	(		Comparação (Not Equal) diferente de (< >)
LT	(		Comparação (Less Than) menor que (<)
LE	(		Comparação (Less or Equal) menor ou igual que (<=)
JMP	C, N	Nome do rótulo	Desvia para o rótulo Nome_do_Rótulo
CAL	C, N	Nome da Função	Invoca a execução de um bloco de funções
RET	C, N		Retorna de uma função ou bloco de função.

Tabela 1 – Principais operadores da linguagem de Lista de Instruções.

5.1 - Operador LD

Mnemônico da palavra inglesa LOAD

- **Operação**: carrega um valor para o acumulador.

- **Modificador**: N
- **Operando**: expressão constante.

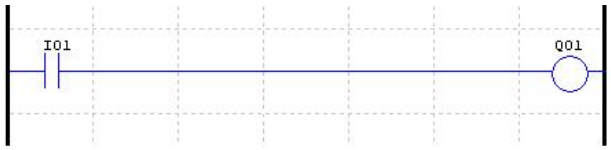
5.2 - Operador ST

Mnemônico da palavra inglesa STORE

- **Operação**: transfere o conteúdo do acumulador para uma variável.
- **Modificador**: N
- **Operando**: variável interna ou de usuário

Exemplo 2: Implemente um programa em Ladder e em Lista de instruções (LI) que tenha a tarefa de acender a lâmpada L sempre que a chave CH fechar.
Solução: Um programa simples no qual a atuação de **uma entrada** causa a atuação de **uma saída**, isto é, utiliza as duas instruções principais que são leitura de variável (**LD**) e atribuição de valor (**ST**), terá o seguinte aspecto , em Ladder e LI.

Ladder



Lista de Instruções	
LD	I1
ST	Q1

Figura 2 – Solução exemplo 2

O processador efetua a leitura de I1 continuamente e executa um programa que atribui o valor lido à saída Q1. Assim, se a chave CH for fechada, fará com que I1 passe ao nível lógico 1, o que vai faze com que L também passe ao nível lógico 1, atuando sobre a saída e, conseqüentemente, acedendo a lâmpada L.

Exemplo 3: Para um contato A do tipo NF, é preciso fazer a leitura de variável negada **LDN**, conforme visto a seguir:

Ladder

Lista de Instruções	
LDN	I1
ST	Q1

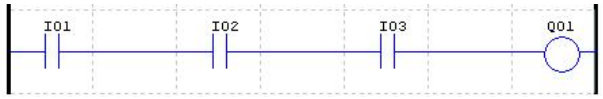
Figura 3 – Solução do exemplo 3.

Neste caso, a partir da instrução **LDN**, o processador efetua a leitura de complemento lógico de **I1** e atribui o valor lido à saída **Q1.**

Exemplo 4: Operação E (AND) – Dada a equação lógica L= I1**.** I2**.** I3, implemente a função lógica no diagrama Ladder e em Lista de Instruções.

Solução:

Ladder



Lista de Instruções	
LDN	I1
AND	I2
AND	I3
ST	Q1

Figura 4 – Solução exemplo 4.

Exemplo 5: Operação OU (OR) – Dada a equação lógica Q1= I1+ I2+ I3□, implemente a função lógica no diagrama Ladder e em Lista de Instruções.

Solução:

Ladder

Lista de Instruções	
LD	I1
OR	I2
ORN	I3
ST	Q1

Figura 5 – Solução exemplo 5.

5.3. Operador S

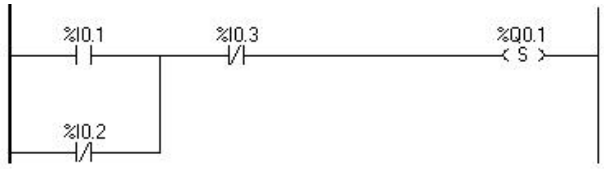
É uma instrução de memorização. A letra S mnemônico da palavra inglesa set.

Operação: Força uma variável booleana a ir para o estado lógico 1 se o acumulador estiver com o valor VERDADEIRO (nível Lógico 1). Nenhuma operação é realizada se o operador estiver com o valor lógico FALSO (nível lógico 0).

- **Modificador**: nenhum.
- **Operando**: variável booleana interna ou de saída

Exemplo 6: Faça o diagrama Ladder e a lista de Instruções correspondentes a dois contatos I1 e I2, NA e NF respectivamente, em paralelo, e um contato I3, NF, em série com ambos. O outro lado do contato I3 está conectado à bobina do tipo set de um relé Q1 de auto retenção.

Ladder



Lista de Instruções	
LD	I1
ORN	I2
ANDN	I3
S	Q1

Figura 6 – Solução exemplo 6

5.4. Operador R

É um opeador que serve para "limpar" o conteúdo da memória. Faz com que o conteúdo da memória vá para o valor zero. A letra R é um mnemônico da palavra inglesa reset.

Operação: força uma variável booleana a ir para o valor lógico 0 se o valor do acumulador for VERDADEIRO (nível lógico 1). Nenhuma operação é realizada se o valor do acumulador for FALSO (nível lógico 0).

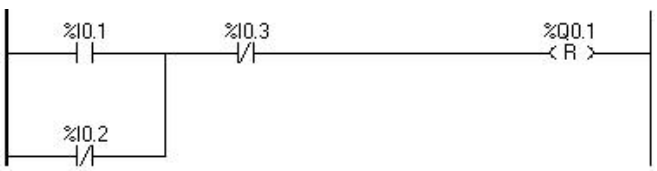
- **Modificador**: Nenhum.
- **Operando**: Variável lógica binária interna ou de saída.

Ps: 1 – A instrução R (reset) existe apenas em programas de CLP e não em um circuito elétrico de chaves ou relés.

2 – A instrução R (reset) sempre trabalha com a instrução S (set) e vice versa.

Exemplo 7: Faça o diagrama Ladder ea lista de Instruções correspondentes a dois contatos I.1 e I.2, NA e NF respectivamente, em paralelo, e um contato I.3, NF, em série com ambos. O outro lado do contato I.3 está conectado à bobina do tipo reset de um relé Q.1 de auto retenção.

Ladder



Lista de Instruções	
LD	I.1
ORN	I.2
ANDN	I.3
R	Q.1

Figura 7 – Solução exemplo 7.

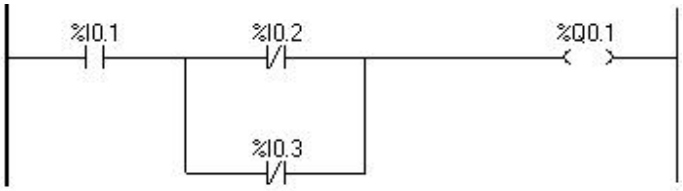
6 – Operações adiadas.

Como a linguagem LI só possui um registrador, certas operações podem ser adiadas para alterar a ordem natural da execução das instruções. Os parênteses são utilizados para representa as operações adiadas.

- "(" = indica que a instrução anterior deve ser adiada;
- ")" = indica que a operação anteriormente adiada deve agora ser executada.

Exemplo 8:

Ladder



Lista de Instruções	
LD	I.1
AND(N	I.2
ORN	I.3
)	
ST	Q.1

Figura 8 – Solução exemplo 8.

Para análise da execução do exemplo fornecido, é preciso definir o conceito de pilha (stack). Para facilitar a compreensão, façamos uma analogia com uma pilha de pratos. Ao empilhar diversos pratos, o último empilhado será o primeiro a ser retirado. Este conceito é também conhecido pela sua abreviação da língua inglesa LIFO – Last Input First Output.

Ao encontrar o modificador "(", o conteúdo do acumulador e o operador são colocados na pilha. Da mesma maneira, quando o operador ")" é encontrado, retira-se o ultimo elemento da pilha e executa-se a operação adiada com o conteúdo atual do acumulador.

É executada da seguinte maneira:

LD %IX1

O conteúdo da entrada %IX1 é transferido para o acumulador
A Instrução seguinte é:

AND (%IX2

Ao encontrar o operador “AND (”, o conteúdo do acumulador e a operação adiada são movidos para a pilha e o operando seguinte (%IX2) é copiado para o acumulador

Então, ao final da segunda linha temos:

- Pilha: %IX1 AND(
- Acumulador: %IX2

Na terceira linha é feita a operação OR %IX3 com o conteúdo atual do acumulador, neste caso %IX2.

Na quarta linha, ao encontrar o operador “)”, é retirado o conteúdo da pilha (%IX1) e executada a operação adiada (AND) com o resultado atual do acumulador. Finalmente o resultado é transferido para a saída pela instrução: ST %QX1.

Existem duas maneiras válidas para implementar a operação adiada do exemplo anterior: carregamento explícito do operador ou forma simplificada.

Num	Descrição/exemplo
1	Carregamento explícito do operador
	AND (N LD %IX1 (nota1) ORN %IX3)
2	Forma simplificada
	AND (N %IX2 ORN %IX3)
Nota 1: No formato 1 o operador LD pode ser modificado ou substituído por outra operação ou chamada de função.	

Figura 9: Duas maneiras de programa instruções adiadas.

Exemplo 9:

Deseja-se avaliar o resultado da seguinte expressão lógica:

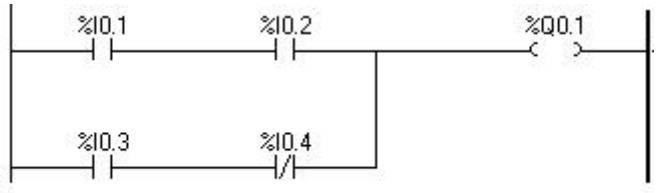
Res:= a1+(a2(a3+a4) a5) + a6

Como pode ser observado, algumas expressões são avaliadas antes das outras, especificadas com a utilização de parêntese para indicar a maior prioridade. Uma implementação em LI pode ser:

```
LD          a1          (*acumulador = a1; *)
                        (*pilha: {}; *)
OR(                     (*OR adiado*)
LD          a2          (*acumulador =a2*)
                        (*pilha: {a1 OR};*)
AND(                    A(*AND adiado*)
LD          a3          (*acumulador = a3*)
                        (*pilha: {a1 OR; A2 AND};*)
OR          a4          (*resultado = a3 OR a4; *)
)                       (*executa o último elemento da pilha (a2 AND)*)
                        (*com o conteúdo atual do acumulador*)
                        (*acumulador = a2 AND (a3 OR a4;*)
                        (*pilha: {a1 OR}*)
AND         a5          (*acumulador = a2 (a3+a4) a5*)
)                       (*executa o ultimo elemento da pilha a1 OR*)
                        (*com o conteúdo atual do acumulador *)
                        (*acumulador = a1+(a2(a3+a4)a5);*)
ST          resultado   (*pilha: {};*)
OR          a6          (*acumulador = a1+(a2(a3+a4)a5)+a6;*)
ST          resultado   (*resultado = acumulador*)
```

Exemplo 10: Dada a equação lógica Q1= (I1**.** I2) + (I3**.** I4□), implemente, em Ladder e em Lista de Instruções.

Ladder



Lista de Instruções	
LD	I.1
AND	I.2
OR(	I.3
ANDN	I.4
)	
ST	Q.1

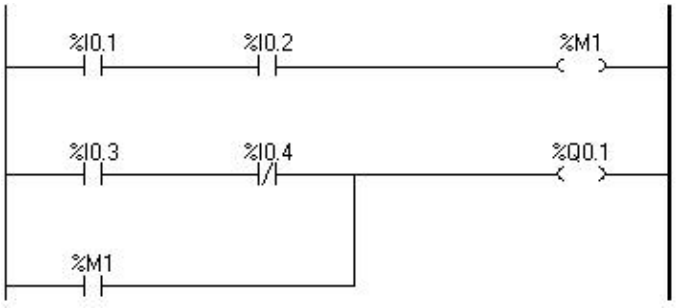
Figura 10: Solução do exemplo 10

Este mesmo programa pode ser reescrito utilizando relés auxiliares, isto é, regiões de memória interna utilizadas para armazenamento temporário de informações identificadas como M_1, M_2, M_3 e

assim sucessivamente. É vantajoso, quanto à clareza do programa, porém desvantajoso em relação ao uso da quantidade de memória.

Desta maneira, o programa anterior pode ser refeito apresentando o mesmo comportamento sob o ponto de vista lógico, conforme ilustra a Figura.

Ladder

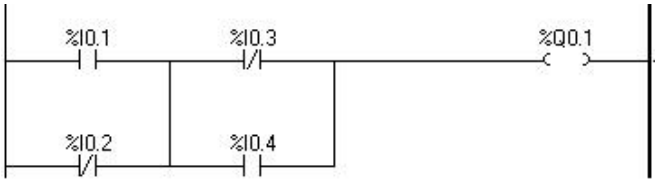


Lista de Instruções	
LD	I1
AND	I2
ST	M1
LD	I3
ANDN	I4
OR	M1
ST	Q1

Figura 11 – Implementação da equação (I1. I2) + (I3. I4□), em Ladder e LI.

Exemplo 11: Dada a equação lógica Q1 = (I1+I2□).(I3□+I4), implemente em Ladder e LI.

Ladder

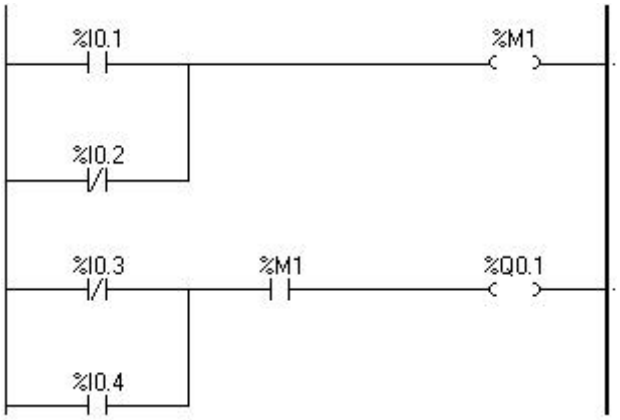


Lista de Instruções	
LD	I1
ORN	I2
AND(N	I3
OR	I4
)	
ST	Q1

Figura 12 – Solução do exemplo 11

A Figura 13 exibe a solução para o exemplo anterior, utilizando relés auxiliares.

Ladder



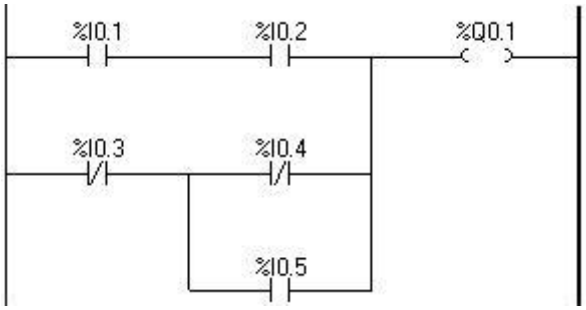
Lista de Instruções	
LD	I1
ORN	I2
ST	M1
LDN	I3
OR	I4
AND	M1
ST	Q1

Figura 13 – Solução do exemplo 11 com o uso de relés auxiliares.

Exemplo 12: Dada a equação lógica Q1 = I1.I2 + I3□□(I4□+I5), implemente em Ladder e LI.

Solução: A figura 14 mostra a solução.

Ladder

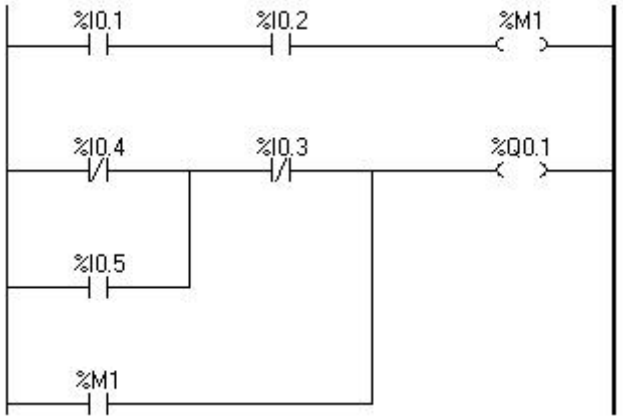


Lista de Instruções	
LD	I1
AND	I2
OR(N	I3
AND(N	I4
OR	I5
)	
)	
ST	Q1

Figura 14 – Solução do exemplo 12.

A figura 15 descreve outra solução para o exemplo 12, agora utilizando relés auxiliares.

Ladder



Lista de Instruções	
LD	I1
AND	I2
ST	M1
LDN	I4
OR	I5
ANDN	I3
OR	M1
ST	Q1

Figura 15 – Solução do exemplo 12 com relés.

7 - Mnemônicos de alguns fabricantes

Antes do surgimento IEC 61131-3, cada fabricante utilizava seu próprio conjunto de mnemônicos. Embora muito parecidos entre si, eram diferentes de um fabricante para outro. Assim, antes de implementar um programa em linguagem de LI em uma aplicação real, deve-se realizar um estudo detalhado do manual do fabricante para determinar os mnemônicos equivalentes à norma IEC 61131-3. A título de exemplo, na tabela 2 são fornecidos os mnemônicos de alguns fabricantes.

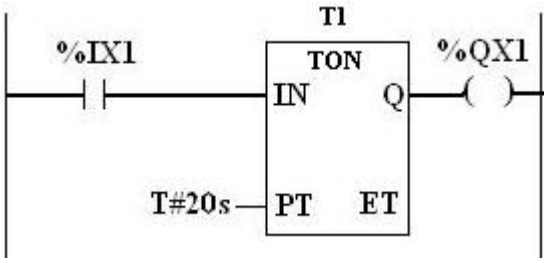
IEC 61131-3	Mitsubishi	OMRON	SIEMENS S7-200
LD	LD	LD	LD
LDN	LDI	LD NOT	LDN
AND	AND	AND	A
ANDN	ANI	AND NOT	AN
OR	OR	OR	O
ORN	ORI	OR NOT	ON
ST	OUT	OUT	=

Tabela 2 – Mnemônicos de alguns fabricantes e seus correspondentes na norma IEC 61131-3

8 – Temporizadores.

A Figura 16 encontra –se a implementação de um temporizador no CLP IPC PS1 (Festo).

Ladder

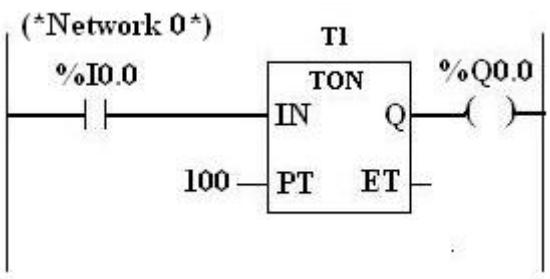


Lista de Instruções
CAL T1 (IN:=%IX1, PT:= T#20s)
LD T1.Q
ST %QX1

Figura 16 – Implementação de um temporizador TON no CLP da Festo IPC PS1 Professional

A figura 17 mostra outra implementação em controlador que segue a norma IEC 61131-3

Ladder

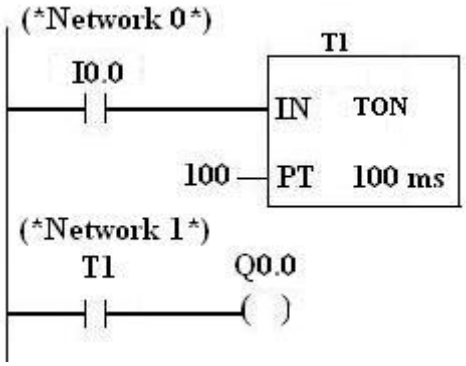


Lista de Instruções
(*Network 0*)
LD %I0.0
TON T1, 100
ST %Q0.0

Figura 17 – Implementação de um temporizador TON no CLP que segue a norma IEC 61131-3

A figura 18 exibe a implementação nos CLPs S7-200 da Siemens.

Ladder

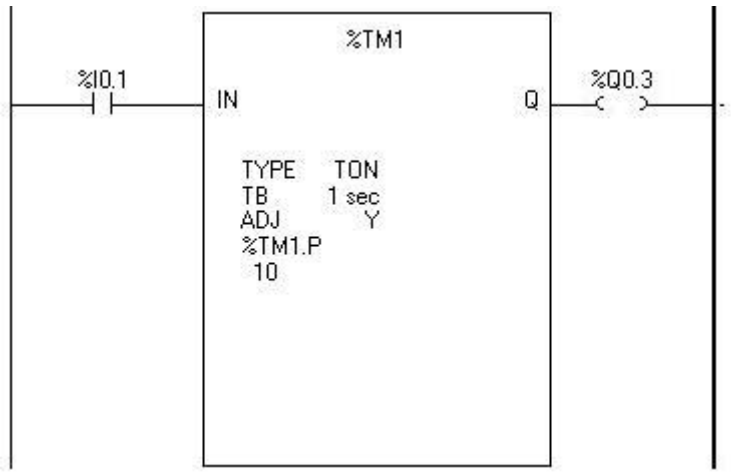


Lista de Instruções
(*Network 0*)
LD I0.0
TON T1, 100
(*Network 1*)
LD T1
= Q0.0

Figura 18– Implementação da função TON no CLP S7-200 da Siemens em Ladder e em LI

A figura 19 exibe a implementação dos CLP's da família Twido

Ladder



Lista de Instruções	
BLK	%TM1
LD	%I0.1
IN	
OUT_BLK	
LD	Q
ST	%Q0.3
END_BLK	

Figura 19 – CLP´s da família Twido – Ladder e Lista de Instrução

As instruções podem conter todos os tipos de operador de entrada padrão,tais como:

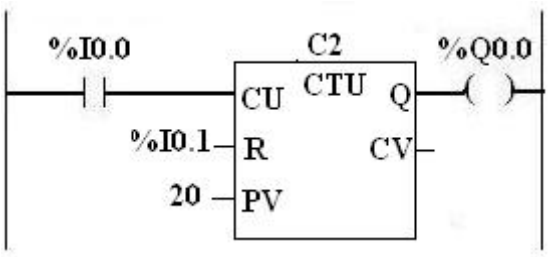
- **TON** (*Timer On Delay*) – atraso associado a execução da lógica de entrada (opção padrão);
- **TOF** (*Timer Off Delay*) - após a execução da lógica de entrada, as saídas correspondentes serão executadas segundo o atraso definido no temporizador (opção não regatilhável)
- **TP** (*Timer Pulse*) - após a execução da lógica de entrada, as saídas correspondentes serão executadas segundo o atraso definido no temporizador (opção regatilhável).

9 – Contadores.

Os blocos podem ser chamados de várias maneiras e os fabricantes têm alguma liberdade de implementação.

Pela norma IEC 61131-3 as funções podem ser chamadas diretamente, sem a necessidade de um operador que as preceda. De fato, o nome da função pode ser considerado um operador. Os parâmetros passados são: o endereço do contador, o contato que está ligado à entrada reset e o valor de PV.

Ladder

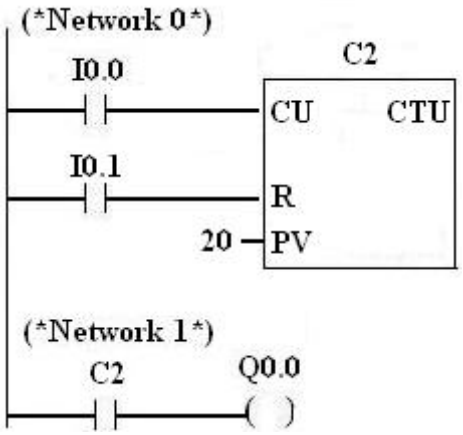


Lista de Instruções	
CAL T1 (IN:=%IX1, PT:= T#20s)	
LD	T1.Q
ST	%QX1

Figura 20 – Implementação de um contador crescente em um CLP que segue a norma IEC 61131-3

Outra forma ainda é seguida pela Siemens na sua linha de CLPs de pequeno porte (S7-200). A figura 21 ilustra uma possível implementação nesses controladores.

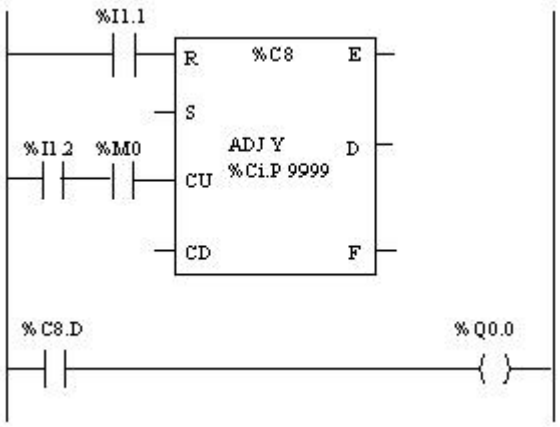
Ladder



Lista de Instruções	
(*Network 0*)	
LD	I0.0
LD	I0.1
CTU	C2, 20
(*Network 1*)	
LD	C2
=	Q0.0

Figura 21 – Implementação de um contador crescente no CLP S7-200 da Siemens

A figura 22 exibe a implementação dos CLP's da família Twido para um contador de ordem crescente.



Lista de Instruções	
BLK	%C8
LD	%I1.1
R	
LD	%I1.2
AND	%M0
CU	
END_BLK	
LD	%C8.D
ST	%Q0.0

Figura 22 – Implementação de um contador crescente no CLP Twido da Schneider.